

## 1.修改购物车类视图

- 目的: 能够编写修改购物车的类视图方法
- 操作流程:
  - 1.类视图中put方法

```
1     def put(self, request):
2         #1, 获取参数
3         dict_data =
4         json.loads(request.body.decode())
5         sku_id = dict_data.get("sku_id")
6         count = dict_data.get("count")
7         selected =
8         dict_data.get("selected", True)
9
10        #2, 校验参数
11        #2.1 为空校验
12        if not all([sku_id, count]):
13            return
14        http.HttpResponseForbidden("参数不全")
15
16        #2.2 sku_id对应的商品对象是否存在
17        try:
18            sku = SKU.objects.get(id=sku_id)
19        except Exception as e:
20            return
21        http.HttpResponseForbidden('商品不存在')
22
23        #2.3 将count整数化
24        try:
25            count = int(count)
26        except Exception as e:
27            return
28        http.HttpResponseForbidden('count数量有误')
29
30        #3, 判断用户状态
31        user = request.user
32        if user.is_authenticated:
33            pass
34        else:
```

## 2.修改购物车(登陆)

- 目的: 能够修改用户redis中的数据
- 操作流程:
  - 1, 获取购物车数据的时候需要带上id

```
123 sku_dict = {
124     "default_image_url": sku.default_image_url.url,
125     "name": sku.name,
126     "price": str(sku.price),
127     "amount": str(sku.price * int(count_selected["count"])),
128     "selected": str(count_selected["selected"]),
129     "count": int(count_selected["count"]),
130     "id": sku_id
131 }
132 sku_list.append(sku_dict)
133
```

- 2.类视图中用户登陆redis数据修改

```
1 class CartView(View):
2     def put(self, request):
3         ...
4         if user.is_authenticated:
5             #3.1 获取redis对象
6             redis_conn =
7             get_redis_connection("cart")
8
9             #3.2 修改数据
10
11             redis_conn.hset("cart_%s"%user.id, sku_id, count)
12
13             if selected:
14
15                 redis_conn.sadd("cart_selected_%s"%user.id, sku_id)
16
17             else:
18
19                 redis_conn.srem("cart_selected_%s" % user.id, sku_id)
20
21             #3.3 拼接数据返回响应
22             context = {
23                 "code": RET.OK,
```

```

19         "cart_sku":{
20             "default_image_url":
sku.default_image_url.url,
21             "name": sku.name,
22             "price": str(sku.price),
23             "amount": str(sku.price *
count),
24             "selected": str(selected),
25             "count": int(count),
26             "id": sku.id
27         }
28     }
29     return http.JsonResponse(context)
30 else:
31     pass
32

```

### 3.修改购物车(未登陆)

- 目的: 能够修改未登录用户的cookie中的数据
- 操作流程:
- 1,类视图,put方法

```

1 def put(self,request):
2     ....
3     else:
4         #4.1 获取cookie中数据
5         cookie_cart =
request.COOKIES.get("cart")
6
7         #4.2 字典转换
8         cookie_dict = {}
9         if cookie_cart:
10             cookie_dict = pickle.loads(
11
base64.b64decode(cookie_cart.encode()))
12
13         #4.3 修改
14         cookie_dict[sku_id] = {
15             "count":count,
16             "selected":selected

```

```

17         }
18
19         #4.4 转换并返回
20         context = {
21             "code": RET.OK,
22             "cart_sku": {
23                 "default_image_url":
sku.default_image_url.url,
24                 "name": sku.name,
25                 "price": str(sku.price),
26                 "amount": str(sku.price *
count),
27                 "selected": selected,
28                 "count": int(count),
29                 "id": sku.id
30             }
31         }
32         response = http.JsonResponse(context)
33         cookie_cart =
base64.b64encode(pickle.dumps(cookie_dict)).decode
()
34
35         response.set_cookie("cart", cookie_cart)
36         return response

```

- 注意点:
  - 响应数据的时候,由于前端没有对selected数据做处理,所以我们需要返回bool类型

## 4.删除购物车(登陆)

- 目的: 能够删除redis中的数据
- 操作流程:
  - 1,类视图delete方法

```

1         def delete(self, request):
2             #1,获取参数
3             dict_data =
json.loads(request.body.decode())
4             sku_id = dict_data.get("sku_id")

```

```

5
6         #2, 校验参数
7         #2.1 为空校验
8         if not sku_id:
9             return
10        http.HTTPResponseForbidden("参数不全")
11
12        #2.2 sku_id对应的商品对象是否存在
13        try:
14            sku = SKU.objects.get(id=sku_id)
15        except Exception as e:
16            return
17        http.HTTPResponseForbidden('商品不存在')
18
19        #3 判断用户登录状态
20        user = request.user
21        if user.is_authenticated:
22            #3.3 获取redis对象
23            redis_conn =
24            get_redis_connection("cart")
25            pipeline = redis_conn.pipeline()
26
27            #3.3 删除数据
28
29            pipeline.hdel("cart_%s"%user.id, sku_id)
30
31            pipeline.srem("cart_selected_%s"%user.id, sku_
32            id)
33
34            pipeline.execute()
35
36            #3.4 返回响应
37            return
38            http.JsonResponse({"code": RET.OK, "errmsg": "suc
39            cess"})
40        else:
41            pass

```

## 5.删除购物车(未登陆)

- 目的: 能够删除cookie中购物车数据

- 操作流程:
  - 1, 类视图中的delete方法

```
1         else:
2             #4.1获取cookie数据
3             cookie_cart =
4             request.COOKIES.get("cart")
5
6             #4.2字典转换
7             cookie_dict = {}
8             if cookie_cart:
9                 cookie_dict =
10                pickle.loads(base64.b64decode(cookie_cart.encode()))
11
12            #4.3删除数据
13            if sku_id in cookie_dict:
14                del cookie_dict[sku_id]
15
16            #4.4返回响应
17            response =
18            http.JsonResponse({"code":RET.OK,"errmsg":"success"})
19
20            cookie_cart =
21            base64.b64encode(pickle.dumps(cookie_dict)).decode()
22
23            response.set_cookie("cart",cookie_cart)
24            return response
```

## 6,全选购物车(登陆)

- 目的: 能够全选reids中的数据
- 操作流程:
  - 1, 子路由(carts/urls.py)

```
1 url(r'^carts/selection/$',views.CartSelectedAllView.as_view()),
```

- 2,类视图(carts/views.py)

```
1 class CartSelectedAllView(View):
2     def put(self, request):
3         #1, 获取参数
4         selected =
5         json.loads(request.body.decode()).get("selected", True)
6
7         #2, 判断用户状态
8         user = request.user
9         if user.is_authenticated:
10            #2.1 获取redis对象, 获取数据
11            redis_conn =
12            get_redis_connection("cart")
13            cart_dict =
14            redis_conn.hgetall("cart_%s"%user.id)
15            sku_id_list = cart_dict.keys()
16
17            #2.2 全选数据
18            if selected:
19
20                redis_conn.sadd("cart_selected_%s"%user.id, *sku_id_list)
21            else:
22
23                redis_conn.srem("cart_selected_%s" % user.id, *sku_id_list)
24
25            #2.3 返回响应
26            return
27            http.JsonResponse({"code": RET.OK, "errmsg": "success"})
28        else:
29            pass
```

## 7, 全选购购物车(未登陆)

- 目的: 全选cookie中的数据
- 操作流程:
- 1, 类视图:

```

1         else:
2             #3.1 获取cookie中的数据
3             cookie_cart =
request.COOKIES.get("cart")
4
5             #3.2 转换字典
6             cookie_dict = {}
7             if cookie_cart:
8                 cookie_dict =
pickle.loads(base64.b64decode(cookie_cart.encode()
))
9
10            #3.3 修改全选状态
11            for sku_id in cookie_dict:
12                cookie_dict[sku_id]["selected"] =
selected
13
14            #3.4 返回响应
15            response =
http.JsonResponse({"code":RET.OK,"errmsg":"success
"})
16                cookie_cart =
base64.b64encode(pickle.dumps(cookie_dict)).decode
()
17
18            response.set_cookie("cart",cookie_cart)
19            return response

```

## 8.合并购物车(美多用户)

- 目的: 能够将cookie的数据合并到redis中
- 操作流程:
  - 1.定义合并方法(carts/utils.py)

```

1 import pickle
2 import base64
3 from django_redis import get_redis_connection

```



```
4 def
merge_cookie_redis_cart(request,user,response)
:
5     """
6     :param request: 为了获取cookie数据
7     :param user: 为了获取redis数据
8     :param response: 为了清空cookie数据
9     :return:
10    """
11    #1,获取cookie数据
12    cookie_cart = request.COOKIES.get("cart")
13
14    #2,判断cookie是否存在,如果有转换
15    if not cookie_cart:
16        return response
17
18    cookie_dict = {}
19    if cookie_cart:
20        cookie_dict =
pickle.loads(base64.b64decode(cookie_cart.encode()))
21
22    #3,合并数据
23    redis_conn = get_redis_connection("cart")
24    for sku_id, count_selected in
cookie_dict.items():
25
26        redis_conn.hset("cart_%s"%user.id,sku_id,count_selected["count"])
27
28        if count_selected["selected"]:
29
30            redis_conn.sadd("cart_selected_%s"%user.id,sku_id)
31
32            else:
33                redis_conn.srem("cart_selected_%s"%user.id, sku_id)
34
35    #4,清除cookie,返回响应
36    response.delete_cookie("cart")
37    return response
```

- 2,登陆地方调用(users/views.py)

```
123  
124  
125  
126  
127 合并方法调用  
128  
129  
request.session.set_expiry(0)  
  
#4, 返回响应  
response = redirect('/')  
response.set_cookie("username", user.username, 3600*24*2)  
response = merge_cookie_redis_cart(request, user, response)  
return response
```

- 注意点:

- 获取购物车页面的时候,如果cookie不存在,需要返回到cart.html

## 9,合并购物车(qq用户)

- 目的: 能够通过qq登陆的用户合并购物车
- 操作流程:(oauth/views.py)
  - qq登陆成功3个位置, 调用merge方法

## 10,购物车简要页面

- 目的: 能够编写购物车获取的类视图
- 操作流程:
  - 1, 子路由

```
1 url(r'^carts/simple/$', views.CartSimpleview.as_  
view()),
```

- 2,类视图

```
1 class CartSimpleview(View):  
2     def get(self, request):  
3         pass
```

## 11,购物车简要页面(登陆用户)

- 目的: 取出redis中的购物车数据
- 操作流程:
  - 1, 类视图

```
1 class CartSimpleView(View):
2     def get(self, request):
3
4         #1,判断用户状态
5         user = request.user
6
7         if user.is_authenticated:
8             #2,1获取redis对象,取出数据
9             redis_conn =
10 get_redis_connection("cart")
11             cart_dict =
12             redis_conn.hgetall("cart_%s"%user.id)
13
14             #2,2拼接数据
15             sku_list = []
16             for sku_id, count in
17 cart_dict.items():
18                 sku =
19 SKU.objects.get(id=sku_id)
20                 sku_dict = {
21                     "id":sku.id,
22                     "name":sku.name,
23
24                     "default_image_url":sku.default_image_url.url
25                 ,
26                     "count":int(count)
27                 }
28                 sku_list.append(sku_dict)
29
30             #2.3返回响应
31             context = {
32                 "cart_skus":sku_list
33             }
34             return http.JsonResponse(context)
35         else:
36             pass
```

- 2.模板渲染

```
56
57
58 <div class="guest_cart fr">
59   <a href="/static/cart.html" class="cart_name fl">我的购物车</a>
60   <div class="goods_count fl" id="show_count">[[cart_total_count]]</div>
61   <ul class="cart_goods_show">
62     {# 遍历购物车数据 #}
63     <li v-for="sku in carts">
64       
65       <h4>[[sku.name]]</h4>
66       <div>[[sku.count]]</div>
67     </li>
68   </ul>
69   {#
```

## 12.购物车简要页面(未登陆用户)

- 目的: 能够展示cookie中的数据,到简要的购物车

- 操作流程:

- 1, 类视图(carts/views.py)

```
1         else:
2             #3.1 获取cookie数据
3             cookie_cart =
4             request.COOKIES.get("cart")
5
6             #3.2字典转换
7             cookie_dict = {}
8             if cookie_cart:
9                 cookie_dict =
10                pickle.loads(base64.b64decode(cookie_cart.encode()))
11
12            #3.3拼接数据
13            sku_list = []
14            for sku_id,count_selected in
15            cookie_dict.items():
16                sku =
17                SKU.objects.get(id=sku_id)
18                sku_dict = {
19                    "id":sku.id,
20                    "name":sku.name,
21
22                    "default_image_url":sku.default_image_url.url
23                },
```

```

18         "count":int(count_selected["count"])
19     }
20     sku_list.append(sku_dict)
21
22     #3.4返回
23     context = {
24         "cart_skus":sku_list
25     }
26     return http.JsonResponse(context)
27

```

## 13,结算订单页展示

- 目的: 能够获取订单结算页面
- 操作流程:

- 1,创建子应用
- 2, 根路由

```

1 url(r'^', include('orders.urls')),

```

- 3,子路由

```

1 url(r'^orders/settlement/$',views.OrderSettleme
ntView.as_view())

```

- 4,类视图

```

1 from django.shortcuts import render
2 from django.views import View
3 from meiduo_mall.utils.my_login_required import
MyLoginRequiredMixin
4
5 class
OrderSettlementView(MyLoginRequiredMixin):
6     def get(self,request):
7         return
render(request,'place_order.html')

```

## 14, 结算订单页展示地址

- 目的: 能够获取用户的收货地址展示
- 操作流程:
  - 1, 类视图

```
1 class
  OrderSettlementView(MyLoginRequiredMixin):
2     def get(self, request):
3
4         #1, 查询用户的地址
5         try:
6             addresses =
request.user.addresses.filter(is_deleted=False
).all()
7         except Exception as e:
8             addresses = None
9
10
11        #2, 拼接数据, 返回响应
12        context = {
13            "addresses": addresses
14        }
15        return
render(request, 'place_order.html', context=context)
16
```

- 2, 模板渲染

```
<dt>寄送到: </dt>
{% for address in addresses %}
<dd><input type="radio" name="address_id" value="{{ address.id }}" checked="">
    {{ address.province }} {{ address.city }}
    {{ address.district }} {{ address.place }} (
    {{ address.receiver }} 收) {{ address.mobile }}
</dd>
{% endfor %}
```